For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**1**	Total no. of pages:	**10**

Declaration of Original Work

By submitting the answer script of this assignment to the subject lecturer through Moodle Centralized Group, you hereby declare that the work in the answer sheet is completely your own work. No part of the answer sheet is taken from other people's work without giving them credit. All references have been clearly cited.

You understand that an infringement of this declaration leaves you subject to disciplinary actions such as mark deduction, disqualification or even expulsion by the College.

If necessary, students may be invited to provide more information on their submission.

(Please refer to the relevant section(s) on plagiarism of the Student Handbook.)

Instructions to Students:

1. Please refer to assignment specification for the submission method
2. Show all your work clearly and neatly. Marks will be deducted for untidy work.

Answer ALL questions.

For answer script - **EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.**

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**2**	Total no. of pages:	**10**

Answer for Question 1

A. Code caption

```
# Q1a)

class Employee():
    # create a constructor
    # with 2 parameters
    def __init__(self, name: str, salary: int):
        self.name = name # initiate the instance variable name
        self.salary = salary # initiate the instance variable salary

    # print an Employee item
    def __str__(self):
        string = self.name + ', ' + str(self.salary)
        return string
```

B. Code caption

```
# Q1b)
# create a list EmpList, with elements mentioned in the question
EmpList = []
```

C. Code caption

```
# Q1c)
# put all the element mentioned in question to EmpList
EmpList.append(Employee('Ada', 15000))
EmpList.append(Employee('Brian', 18000))
EmpList.append(Employee('Carson', 12000))
EmpList.append(Employee('Dave', 14000))
```

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**3**	Total no. of pages:	**10**

Answer for Question 2

A. Code caption

```python
# Q1a)
class Employee():
    # create a constructor
    # with 2 parameters
    def __init__(self, name: str, salary: int):
        self.name = name # initiate the instance variable name
        self.salary = salary # initiate the instance variable salary

    # print an Employee item
    def __str__(self):
        string = self.name + ', ' + str(self.salary)
        return string

    # Q2a)  A method insertionSort to accept Employee list
    # and sort the list by using their salary
    def insertionSort(others: list):
        for i in range(1, len(others)):
            key = others[i]

            j = i - 1
            # print(key)

            while j >= 0 and key.salary < others[j].salary:
                others[j + 1] = others[j]
                j -= 1
            others[j + 1] = key
```

B. Code caption

```python
# Q2b) Call insertionSort method
#   to sort the list by using employee's salary
Employee.insertionSort(EmpList)
```

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**4**	Total no. of pages:	**10**

C. Code caption

```
#Q2c)
# Print the information of each Employee before and after sorting
print('Before Insertion Sort: ')
for item in EmpList:
    print(item)
# Before Insertion Sort:
# Ada,15000
# Brian,18000
# Carson,12000
# Dave,14000


print('\nAfter Insertion Sort: ')
# Q2b) Call insertionSort method
#   to sort the list by using employee's salary
Employee.insertionSort(EmpList)
for item in EmpList:
    print(item)
# After Insertion Sort:
# Carson,12000
# Dave,14000
# Ada,15000
# Brian,18000
```

Program execution result:

```
Before Insertion Sort:
Ada, 15000
Brian, 18000
Carson, 12000
Dave, 14000

After Insertion Sort:
Carson, 12000
Dave, 14000
Ada, 15000
Brian, 18000
```

More test cases:

```
# test case 1
case1 = [
    Employee('Carson', 12000),
    Employee('Ada', 15000),
    Employee('Brian', 18000),
]
```

```
#test case 2
case2 = [
    Employee('Brian', 18000),
    Employee('Ada', 15000),
    Employee('Dave', 13000),
]
```

```
# test case 3
case3 = [
    Employee('Ada', 15000),
    Employee('Carson', 12000),
    Employee('Brian', 15000),
    Employee('Dave', 11000),
]
```

For answer script - **EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.**

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**6**	Total no. of pages:	**10**

Printing information before and after sort:

```
# print out all unsorted case 1 to 3
print('\nBefore Insertion Sort, case 1: ')
for item in EmpList:
    print(item)

print('\nBefore Insertion Sort, case 2: ')
for item in EmpList:
    print(item)

print('\nBefore Insertion Sort, case 3: ')
for item in EmpList:
    print(item)


# print out all sorted case 1 to 3
print('\nAfter Insertion Sort case 1: ')
Employee.insertionSort(case1)
for item in case1:
    print(item)


print('\nAfter Insertion Sort, case 2: ')
Employee.insertionSort(case2)
for item in case2:
    print(item)


print('\nAfter Insertion Sort, case 3: ')
Employee.insertionSort(case3)
for item in case3:
    print(item)
```

Result:

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**7**	Total no. of pages:	**10**

```
Before Insertion Sort, case 1:
Carson, 12000
Dave, 14000
Ada, 15000
Brian, 18000

Before Insertion Sort, case 2:
Carson, 12000
Dave, 14000
Ada, 15000
Brian, 18000

Before Insertion Sort, case 3:
Carson, 12000
Dave, 14000
Ada, 15000
Brian, 18000

After Insertion Sort case 1:
Carson, 12000
Ada, 15000
Brian, 18000

After Insertion Sort, case 2:
Dave, 13000
Ada, 15000
Brian, 18000

After Insertion Sort, case 3:
Dave, 11000
Carson, 12000
Ada, 15000
Brian, 15000
```

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**8**	Total no. of pages:	**10**

Answer for Question 3

A. In the aspect of efficiency, bubble sort has the average time complexity of $O(n^2)$, but quick sort has the average time complexity of O(n log n), which is faster than bubble sort.

The quick sort applies the idea of pivot, partition, and recursion, it based on partitioning of array of data into smaller arrays. But in bubble sort, it partially sorts some continuous elements in each iteration, we usually called them as bubbles.

The quick sort requires extra coding effort on implementation, because of the partitioning. But bubble sort does not, it only required 2 for-loop to directly sort the data array.

B. Quick sort is based on recursion

Partitioning pseudo code:

```
function partitioning( List, starting, ending){
        x = List[starting]
        i = starting
        for j = starting + 1 to ending{
                if  List[j] < x then{
                        i = i + 1
                        swap(List[i], List[j])
                }
        }
        swap(List[i], List[starting])
        return i
}
```

Sorting pseudo code:

```
function quicksort(List, starting, ending){
        if  starting >= ending
                return
        else{
                pivot = partitioning(List, starting, ending)
                quicksort(List, left, pivot - 1)
                quicksort(List, pivot +  1, ending)
        }
}
```

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**9**	Total no. of pages:	**10**

Demonstration of the quick sort algorithm:
unsorted = [1,9,2,8,7,3]
quicksort(unsorted, 0, 5)

1st pass:
pivot = partitioning(unsorted, 0, 5)
list == [1, 9, 2, 8, 7, 3]
pivot = 0
quicksort(unsorted, 0, -1) ➔ return
quicksort(unsorted, 1, 5) ➔ 1st inner quicksort() (right hand side):
pivot = partitioning(unsorted, 1, 5)
list == [1, 3, 2, 8, 7, 9]
pivot = 5
quicksort(unsorted, 1, 4) ➔ 2nd inner quicksort() (left hand side):
pivot = partitioning(unsorted, 1, 4)
list == [1, 2, 3, 8, 7, 9]
pivot = 2
quicksort(unsorted, 1, 1) ➔ return
quicksort(unsorted, 3, 4) ➔ 3rd inner quicksort() (right hand side):
pivot = partitioning(unsorted, 3, 4)
list == [1, 2, 3, 7, 8, 9]
pivot = 4
quicksort(unsorted, 3, 3) ➔ return
quicksort(unsorted, 5, 4) ➔ return
(as above 2 inner quicksort() is returned, the list is sorted)
quicksort(unsorted, 6, 5) ➔ return

For answer script - EVERY answer sheet MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group and (e) page number.

Name:	Tsoi Yiu Chik	Student No.:	20195601
Subject Code	SEHH2239	Subject Lecture Group:	201B
Page No.:	**10**	Total no. of pages:	**10**

Answer for Question 4

Value a to f: 20, 1, 19, 95, 56, 60

A. Initial state:
20, 1, 19, 95, 56, 60
1st pass:
1, 20, 19, 95, 56, 60 ← swapped 20, 1
1, 19, 20, 95, 56, 60 ← swapped 20, 19
1, 19, 20, 95, 56, 60 ← 20 < 95, no swap
1, 19, 20, 56, 95, 60 ← swapped 95, 56
1, 19, 20, 56, 60, 95 ← swapped 95, 60
2nd pass, no swap:
1, 19, 20, 56, 60, 95 ← 1<19, no swap
1, 19, 20, 56, 60, 95 ← 19<20, no swap
1, 19, 20, 56, 60, 95 ← 20<56, no swap
1, 19, 20, 56, 60, 95 ← 56<60, no swap
1, 19, 20, 56, 60, 95 ← 60<95, no swap

The sorting is early terminated in the case.
Reason:
In the 2nd pass, there is no swap between, 1 and 19, 19 and 20, 20 and 56, 56 and 60. Therefore, all elements in the remaining sub-list are sorted, which also means the whole list is already sorted (because the largest number is already at the last), so early termination exists.

B. Sort 20, 1, 19, 95, 56, 60
Start with 20
20

insert 1 ➔ 1,20
__,20 ➔ 1,20

insert 19 ➔ 1, 19, 20
1, __,20 ➔ 1, 19, 20

insert 95 ➔ 1, 19, 20, 95
1, 19, 20, __ ➔ 1, 19, 20, 95

insert 56 ➔ 1, 19, 20, 56, 95
1, 19, 20, __, 95 ➔ 1, 19, 20, 56, 95

insert 60 ➔ 1, 19, 20, 56, 60, 95
1, 19, 20, 56, __, 95 ➔ 1, 19, 20, 56, 60, 95